Bolt Beranek and Newman Inc.

12

Report No. 4609

Combined Quarterly Technical Report No. 20

SATNET Development and Operation
Pluribus Satellite IMP Development
Remote Site Maintenance
Internet Development
Mobile Access Terminal Network
TCP for the HP3000
TCP-TAC
TCP for VAX-UNIX

February 1981

Prepared for:
Defense Advanced Research Projects Agency

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A096 114	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) COMBINED QUARTERLY TECHNICAL REPORT No. 20		5. TYPE OF REPORT & PERIOD COVERED 11/1/80 to 1/31/81
		6. PERFORMING ORG. REPORT NUMBER 4609
7. AUTHOR(s) R. D. Bressler		8. CONTRACT OR GRANT NUMBER(s) MDA903-80-C-0353, & 0214 N00039-78-C-0405 N00039-79-C-0386 N00039-80-C-0664
9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman Inc. 50 Moulton Street Cambridge, MA 02238		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA Order Nos. 3214 and 3175.17
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209		12. REPORT DATE February 1981
		13. NUMBER OF PAGES 84
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) DSSW Rm. 1D, The Pentagon Washington, DC 20310	NAVELEX Washington, DC 20360	15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE/DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer networks, packets, packet broadcast, satellite communication, gateways, Transmission Control Program, UNIX, Pluribus Satellite IMP, Remote Site Module, Remote Site Maintenance, shipboard communications, Terminal Access Controller, VAX.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This Quarterly Technical Report describes work on the development of and experimentation with packet broadcast by satellite; on development of Pluribus Satellite IMPs; on a study of the technology of Remote Site Maintenance; on the development of Inter-network monitoring; on shipboard satellite communications; and on the development of Transmission control protocols for the HP3000, TAC, and VAX-UNIX.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

[illegible]

(14) BBN-

Report No. 4609

Bolt Beranek and Newman Inc.

Number

(6)

COMBINED QUARTERLY TECHNICAL REPORT NO. 20

SATNET DEVELOPMENT AND OPERATION,
PLURIBUS SATELLITE IMP DEVELOPMENT,
REMOTE SITE MAINTENANCE,
INTERNET DEVELOPMENT,
MOBILE ACCESS TERMINAL NETWORK,
TCP FOR THE HP3000
TCP-TAC,
TCP FOR VAX-UNIX,

(9) Rept. for 1 Nov 80-31 Jan 81

(11) Feb~~ruary 19~~81

(12) 99

(10) [illegible] Blumenthal

This research was supported by the Defense Advanced Research Projects Agency under the following contracts:

(15) N00039-78-C-0405, ARPA Order No. 3175.17
N00039-79-C-0386
MDA903-80-C-0353, ARPA Order No. 3214
MDA903-80-C-0214, ARPA Order No. 3214
N00039-80-C-0664

Submitted to:

Director
Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, VA 22209

Attention: Program Management

1B

Table of Contents

TABLES

1 INTRODUCTION

This Quarterly Technical Report is the current edition in a series of reports which describe the work being performed at BBN in fulfillment of several ARPA work statements. This QTR covers work on several ARPA-sponsored projects including (1) development and operation of the SATNET satellite network; (2) development of the Pluribus Satellite IMP; (3) Remote Site Maintenance activities; (4) inter-network monitoring; (5) development of the Mobile Access Terminal Network; (6) TCP for the HP3000; (7) TCP-TAC; and (8) TCP for the VAX-UNIX. This work is described in this single Quarterly Technical Report with the permission of the Defense Advanced Research Projects Agency. Some of this work is a continuation of efforts previously reported on under contracts DAHC15-69-C-0179, F08606-73-C-0027, F08606-75-C-0032, MDA903-76-C-0213, and MDA903-76-C-0252.

2 SATNET DEVELOPMENT AND OPERATION

2.1 Host-SATNET Protocol Deadlock

An inadequacy in the Host-SATNET protocol to guard against the loss of a single Restart Complete message is believed to have been the cause of communications deadlocks between the UCL Gateway and the Goonhilly Satellite IMP on several occasions. Since we do not expect messages to be lost in the circuit due to the VDH link protocol layer beneath the Host-SATNET protocol layer, presumably a message was lost in one of the machines unbeknown to either one. The inadequacy in the protocol is revealed by the following scenario.

- Consider two sites {A} and {B} using Host-SATNET protocol to communicate with one another.
- {A} sends a Restart Request message and listens for a Restart Complete message.
- {B} receives the Restart Request sent by {A}, sends a Restart Complete, and listens for an answering Restart Complete. If {B} fails to receive a Restart Complete, it continues sending Restart Completes at ten second intervals.
- {A} receives the first Restart Complete, answers with a single Restart Complete, and enters the "Device-Access-Up" state.

If the Restart Complete sent by {A} is not received by {B}, {B} is prevented from entering the "Device-Access-Up" state. According to the current protocol, {B} must keep sending Restart Completes until it receives either a Restart Complete or another Restart Request. {A}, on the other hand, is in the "Device-

Access-Up" state, and so will discard any Restart Completes as spurious. Neither will it send a Restart Request, for it is still hearing {B}'s Restart Complete messages, signifying an active host and precluding the host time-outs.

There are two obvious solutions to this problem. First, each side can limit the number of consecutive Restart Completes sent. After sending some specific number on the order of 10 consecutive Restart Completes, the site can infer that something is wrong and send a Restart Request. (In the scenario above, {B} would send 10 Restart Completes followed by a Restart Request.) Second, each side can detect device-access problems by ignoring Restart Complete and Restart Request messages when determining when the other side was last heard. (In the scenario above, {A} would detect that {B} has not sent anything meaningful in a long time and would send a second Restart Request.) Note that sending a Restart Complete in response to receipt of every Restart Complete is not a solution, since this leads to a never-ending sequence of Restart Completes. Because we believe corrective action from a site which has not entered the "Device-Access-Up" state is preferable, we implemented the first solution in the Satellite IMP.

2.2 Software Problems Fixed

We released Satellite IMP version 3.3.10, which incorporates the software changes described below.

When the installation of PSP terminals into all the Satellite IMP sites located at INTELSAT earth stations was completed, we set as the default state the enabling of the transfer of T&M data from the PSP terminal to the Satellite IMP. This is accomplished through the insertion of the appropriate commands into the sequence of table-driven commands sent to the PSP terminal upon Satellite IMP initialization.

Subsequently, we noticed that the Satellite IMP would not fall gracefully into its co-resident Satellite Loader/Dumper program during instances when continued Satellite IMP program operation was proscribed due to inconsistent state information or when commanded by the SATNET Monitoring and Control Center (SMCC). Recovery required manual resetting of the PSP terminal by site personnel. The cause of the problem was traced to the Satellite Loader/Dumper program rejecting any received packets with invalid hardware checksums, as happens when the PSP terminal appends T&M information. Rather than change the Satellite Loader/Dumper program to recognize these packets, we inserted into its initialization routine the command to reset the PSP terminal; hence, the appending of T&M data onto received packets is now terminated upon entry into the Satellite Loader/Dumper program.

Once the appending of T&M data is enabled, large numbers of dummy packets appear, being generated within the PSP terminal and sent to the adjacent Satellite IMP. To provide more information on the situation, we modified the Satellite IMP program to send trap reports to the SMCC upon receipt of dummy packets. Included in the trap report was the value of the PSP status word (fourth T&M word) for subsequent determination of the cause of the dummy packets. After receiving copious trap reports, we deimplemented dummy packet trap reports and implemented internal counters of dummy packets instead.

A number of traps generated by the Satellite IMP and identified solely by the program counter value were converted to "invariant" traps, so as to provide an accompanying prin message in MONITOR and QUERY trap reports.

The TTY fake host was modified to provide a more dependable mechanism for switching between the hardware loopback test and the resident DDT. Modifications include:

- The character CTRL-@ (CTRL-SHIFT-P) resets the TTY fake host to its default state; field sites reset to the command level of the hardware loopback test, while the BBN backroom sites reset to the local DDT.

- The character CTRL-C resets the TTY fake host to the command level of the hardware loopback test. This command provides the BBN backroom sites their only entry to the hardware loopback test.

- While Sense Switch 4 is set on the Honeywell 316, DDT messages are inhibited from printing on the console TTY. This mecnanism is provided as an escape from a long and unwanted printout. Note, however, that DDT is still active and will execute any commands typed without printing out the answer.

2.3 Hardware Problems Fixed

Below are summarized several hardware problems which were manifested in the operation of SATNET during the last quarter. In cases involving the Honeywell 316, we not only diagnosed the problems, but also corrected them. In other cases, we were involved primarily in the detection of the problems and helped with diagnosis.

The substantial performance problems with the SATNET channel reported on in the last Quarterly Technical Report have largely disappeared. Beginning 6 November 1980, SATNET channel operations have markedly improved; channel packet error rates dropped over an order of magnitude, contributing to increased stability of the network. Neither COMSAT nor BBN has been able to unearth the reason for the dramatic improvement; possibly an external interfering source was removed.

The transition late last spring of the NORSAR-SDAC ARPANET trunking circuit from a commercial satellite circuit to a military-controlled satellite circuit caused severe deterioration in service and availability to users of the NORSAR TIP. Currently the deterioration is still present, but to a lesser extent. During outages of this circuit, the alternate monitoring and control path to SATNET is unavailable, leaving the BBN gateway to Etam Satellite IMP circuit as the only monitoring and control path. During one particularly long outage lasting over a week, we converted the ARPANET trunking circuit via SATNET from providing London TIP connectivity to providing NORSAR TIP connectivity, so that the ARPANET Network Control Center (NCC) could check various operations necessary for coordinating circuit repairs. Note that it is impossible to have both the London TIP and the NORSAR TIP supported simultaneously, since the ARPANET IMP program is unable to multiplex two IMPs on a single port.

Outages of the Goonhilly Satellite IMP during the last quarter were traced to several problems. On one occasion, after we determined that the Satellite IMP program was still running correctly, COMSAT diagnosed that the 15-volt power supply in the PSP terminal had failed. To restore operation, site personnel connected an external laboratory power supply to the equipment.

Later, a new PSP terminal power supply was sent by COMSAT to replace the laboratory power supply.

On another occasion, a power glitch forced the Honeywell 316 into its TTY paper tape bootstrap loader and destroyed the contents of memory locations 1 and 2 in the bootstrap loader. Since site personnel under guidance from the SMCC were unable to restore locations 1 and 2 from the front panel, which is the only mechanism available for loading the contents of memory locations 1 through 17, we believed that the problem was due to dirty contacts on the front panel switches. Fortunately, the Satellite IMP program was intact, and we were able to continue operation by restarting the program. At the following preventive maintenance session, it was determined that the problem was due to confusion on the part of site personnel; nevertheless, the front panel switches were thoroughly cleaned. Also the 12-volt voltage regulator card was adjusted, and a noisy fan on the 9600 baud Codex modem at Goonhilly was replaced.

We believe an intermittent hardware problem in the 9.6 Kb/s Codex modem either at Goonhilly or at London caused the ARPANET trunking circuit via SATNET to become one-way on several occasions. It appears that when the circuit was trying to come up, none of the SDAC IMP packets arrived at the London TIP, inasmuch as the London TIP generated no acknowledgements to the SDAC IMP routing messages. At the same time the SMCC indicated traffic arriving satisfactorily from London. Typically, looping and unlooping the modem restores correct operation.

Several outages with the Tanum Honeywell 316 Satellite IMP were traced to a malfunction in the power-fail restart circuit. As a temporary measure, the circuit was disabled by site personnel; later, the 12-volt voltage regulator card was adjusted, and the circuit reenabled.

For several days, a cable fault in the 50 Kb/s landline circuit between the SDAC IMP and the ETAM Satellite IMP forced us to re-route the ARPANET trunking circuit via SATNET to the NORSAR TIP rather than to the SDAC IMP for providing London TIP connectivity. Normal service was restored after repairs were effected by WUI.

3 PLURIBUS SATELLITE IMP DEVELOPMENT

The major activity during the past quarter continued to be support of on-going integration of the PSAT with other elements of the Wideband Packet Satellite Network. Other technical efforts were related to continuing work on the Satellite Modem Interface (SMI), initiation of performance measurement/enhancement activities, and refining the definition and implementation of the Host Access Protocol (HAP).

Wideband Network integration activities have involved working in concert with both Linkabit and Western Union. A major milestone was reached during the Wideband Network contractors meeting at ISI (November 6 and 7). At that time, the ISI PSAT was able to loop datagram packets to itself over the satellite via the ESI and Earth Terminal for the first time. In mid-December, the second site at Lincoln Labs was brought up and tested, communicating with itself over the Westar channel. An attempt at communicating between the East and West coasts was not successful at that time due to a variety of problems related to incompatible power levels at the two sites and incompatible initial acquisition sequences for the PSAT and ESI. These problems were addressed during the later part of December and early January in preparation for subsequent periods of scheduled satellite access. The initial acquisition procedure developed for use with the prototype ESIs is described in section 3.1

below. This procedure may need to be modified when operational status of the Wideband Network with the ADM ESIs is achieved.

During the January integration tests, the Lincoln Labs PSAT was able to lock onto the leader packets transmitted by the ISI PSAT for about 5 seconds and represented the first successful (but brief) instance of cross-country PSAT-to-PSAT transmission. Because this test was carried out with the ISI site unattended, it was not possible to determine the precise cause of the interruption of service. Our next opportunity to test on the Westar channel will locate Linkabit personnel at the ISI site so that a better understanding of the power levels on the channel can be determined. Our previous problems in maintaining communication over the satellite channel may also have been due to a bug in the SMI hardware (described below) which has now been eliminated.

The work on the SMI carried out during the quarter involved continued debugging of the design and creation of a documentation and support package that will facilitate manufacture of additional boards without involvement of the design team. Based on our integration activities with the ESI over the past several months, as well as continuing in-house testing in a looped environment, a number of changes have been made to the original SMI design to remove design flaws. Significant effort during the quarter went into upgrading programmer documentation, parts lists, block diagrams, functional descriptions, production drawings, and wire wrap tapes to reflect these changes. In addition, a stand-alone software program was completed which will serve both as a diagnostic tool for the field and a test program for use during the manufacture and repair of SMI boards. This diagnostic/test program for the SMI applies about 100 distinct tests in order to validate the operation of the interface or identify the subsystem which is causing erroneous behavior. It was this software, in fact, which during the later part of the quarter identified a problem with the checksum logic on the MSR and MSX cards which had gone undetected for several months. This problem resulted in all packets, whether corrupted or not, having their hardware checksums marked as good by the SMI. This design flaw was subsequently corrected and SMIs already in the field were retrofitted to reflect the fix. A summary of the tests carried out by the SMI test software package is contained in section 3.2 below.

One of BBN's major tasks this year is PSAT performance measurement and enhancement. Work on this task was initiated during the quarter. SMI latency problems associated with the speed of the existing poller processor have already been documented. We began work during the quarter on development of the SuperSUE poller, an intelligent microprogrammed DMA controller designed to address this problem. To date, work has focused on understanding the operation of the augmented 2901 microengine that will be programmed with the poller loop. Measurements made during the quarter have pointed out another area where "tuning" can probably reduce the SMI latency problem. DMA transfers to buffers on the memory bus (which also supports modules containing code pages) can apparently experience excessive delays due to memory bus loading. The solution to this problem appears to be to separate the buffers from the code and variables pages and put them on their own memory bus. This will require eliminating the spare PSAT code pages, but such elimination will probably be necessary in any case to add additional buffers to the PSAT system. We plan to make this rearrangement in the PSAT memory allocation in the near future and to evaluate its impact on processing power as well as its impact on decreasing SMI latency.

Related to the issue of PSAT performance is the area of Wideband Network performance. BBN has been concerned for some time about the impact of the current PSAT/ESI stateword protocol on the performance of the system. Our feeling is that the current definition which supports up to 15 statewords is inadequate to support the traffic that will exist in the Wideband Network over the next several years. During the quarter we attempted to organize our thoughts on this subject and develop a recommended alternative to the current design. Section 3.3 below summarizes our thoughts on this subject.

Finally, during the quarter we continued work on the development of the Host Access Protocol (HAP). We expanded the existing HAP implementation in a number of minor ways to achieve compatibility with the interface specification document (W-Note 17) distributed at the Wideband Network contractors meeting at ISI in November. We also met with J. Forgie from Lincoln Labs to discuss possible improvements/changes to W-Note 17.

3.1 PSAT/ESI Initial Acquisition Sequence

The PSAT/ESI interface specification contained in W-Note 13 defines the messages passed between the PSAT and ESI but does not precisely specify the sequence of exchanges necessary to establish communication on the satellite channel. As part of the system integration and test activity between BBN and Linkabit, a "coming up" procedure has been developed. This procedure is oriented specifically toward operation with the prototype ESIs that are being deployed in the field on an interim basis. It is possible that these procedures may need to be modified when the ADM ESIs become available during the second quarter of CY81. The diagram below illustrates the currently defined procedure to be followed by a leader PSAT which is coming up on the channel.

Leader (Prototype)

```
        PSAT                          ESI
        ----                          ---

        Power Up                      Power Up
          .                             .
          .                             .

 1.     Clear SMI, (RESET Pulse)------->  |
                                          |
 2.          ------SET PARAMETER WORDS->  |
                                          |GFO
 3.     Enable Poller                     |Acquisition
                                          |(Old
 4.     Wait for GFO Response             | Parameters)
          .                               |
          .                               |
             <---------GFO COMPLETE------ V
                     (No Detect)     Process
                                  SET PARAMETER WORDS

 5.          ---------ACQUIRE GFO------>  |
                                          |
 6.      Wait for GFO Response            |GFO
           .                              |Acquisition
           .                              |(New
           .                              | Parameters)
              <---------GFO COMPLETE----- V
                      (No Detect)

 7.           --------ACQUIRE GFO------>  |
                                          |
 8.      Enable CPM                       |
                                          |
 9.           --------Leader Packet---->  |GFO
                          .               |Acquisition
                          .               |(New
              --------Leader Packet---->  |Parameters)
                          .               |
                          .               |
              <-------Leader Packet-----  |
              <-------GFO COMPLETE (ok)-  V

 10. Calculate Range

 11. PSAT Up
```

The first attempt by the ESI at GFO Acquisition is automatic based on receipt of the RESET signal from the PSAT. The second GFO Acquisition is an attempt to acquire with the new parameters (e.g. symbol rate) passed by the PSAT to the ESI in a SET PARAMETER WORDS local control packet. Since the PSAT in question is the leader, it will hear nothing on the channel and fail to acquire this second time as well. The third attempt at GFO Acquisition is successful since the ESI will hear the leader packets being transmitted by the PSAT.

In the case of a non-leader PSAT, the coming up sequence is slightly different, as illustrated below.

Non-Leader(Prototype)

```
          PSAT                          ESI
          ----                          ---
            +--                       --+
            |                           |
            |       Steps 1-4           |
            |                           |
            |     Same as Leader        |
            |                           |
            +--                       --+

             <-------GFO COMPLETE-------
                 If "No Detect" Step 5
                        else
                 If "Ok" Step 7

 5.          ---------ACQUIRE GFO------->

 6.       Wait for GFO Response           |GFO
            .                             |Acquisition
            .                             |(New
            .                             V Parameters)
              <--------GFO COMPLETE-----
                          (ok)

 7.       Enable CPM

 8.       Calculate Range

 9.       PSAT Up
```

In this case, the scenario is somewhat simpler since the ESI is able to acquire GFO based on ongoing leader transmissions. Only two attempts at GFO acquisition are required, therefore.

3.2 Satellite Modem Interface Test Software

The Satellite Modem Interface test software is a stand-alone program which runs on the Pluribus multiprocessor and serves both as a diagnostic tool for use in the field and a test program to be used for both initial fabrication and repair of SMI cards. The software will run on any Pluribus configuration with 8K words of local memory and 8K words of common memory although it is expected that it will be primarily used on larger configurations which also run the PSAT application program.

The test software consists of about 100 distinct tests. The user of the software is allowed to specify a starting test number, a concluding test number, and a number of times that each test is to be performed. The user is informed as to the progress of the test sequence via both the front panel lights and messages displayed on the CRT. When a test fails, the user is informed (1) where the failure occurred (i.e. what test was in progress), (2) what information was expected, and (3) what information was actually detected. This can be used to identify the specific portion of the interface which is malfunctioning in many cases. The following list summarizes the test classes currently available. The software is designed in a modular fashion to permit the easy addition of new tests at a later date.

DEVICE REGISTER TESTS (Tests 1-5) - These tests exercise the device control registers, checking that bits can be set and cleared and that no QUITs are generated when the device registers are referenced.

CLOCK TESTS (Tests 101-109) - These tests verify operation of the local clock and watchdog-timer. The clock frequency as well as the toggling of all the clock bits are tested. The ability to set and reset the CLOCK-READ-ERROR bit is also checked.

SINGLE PACKET OUTPUT TESTS (Tests 201-208) - These tests verify operation of the transmit side of the SMI for single buffer single packet bursts. Status register bits are examined to determine if the DMA operation and data transmission both start and complete properly.

MULTI-BUFFER SINGLE PACKET OUTPUT TESTS (Tests 301-30E) - These tests verify operation of the output side of the interface for multi-buffer single packet bursts. They are similar in detail to the single packet output tests described above.

MULTI-PACKET OUTPUT TESTS (Tests 401-40A) - These tests verify operation of the transmit side of the SMI for multi-packet bursts. These tests are similar in detail to the single packet and multi-buffer output tests described above.

TRANSMIT PROTECT TIMER TESTS (Tests 501-503) - These tests exercise the transmit protect timer to verify that it fires within a specified time interval.

OUTPUT FLUSHING TESTS (Tests 601-606) - These tests exercise the transmit output flushing mechanism. They verify that buffers of a packet beyond the one causing the flushing state are discarded and that subsequent bursts are processed normally.

TRANSMIT & RECEIVE RESPONSE TO QUITS TESTS (Tests 701-702) - These tests check QUIT handling on the transmit and receive sides of the SMI by specifying a buffer in non-existent memory.

SINGLE PACKET TRANSMIT/RECEIVE TESTS (801-80A) - These tests check both the transmit and receive sides of the SMI by transmitting and receiving a single packet single-buffer burst. Status registers and buffer end pointers are examined to verify that the transmit and receive operations start and complete properly. In addition, a series of burst validity tests are performed (see F001-F007 below).

MULTI-BUFFER SINGLE PACKET TRANSMIT/RECEIVE TESTS (901-90D) - These tests transmit and receive a single packet multi-buffer burst. They are similar in detail to the single buffer transmit/receive tests described above.

CHECKSUM TESTS (A01-A02) - These tests validate the transmit and receive checksum logic. The hardware checksum is actually compared to a software computed checksum to verify that it is being computed properly. Packets with bad checksums are generated to make sure that they are detected properly by the receive logic.

WORD COUNT TESTS (C01) - These tests exercise the bits in the word count field by sending packets of increasing length, each having a single one bit in the word count field and the remaining bits zero.

TRANSMIT TIME TESTS (D01) - These tests exercise the bits in the transmit clock by sending packets at different times, each having a single clock bit one and the remaining clock bits zero.

COMMON BURST VALIDITY TESTS (F001-F007,F100) - These tests are performed whenever a burst is both transmitted and received. In addition to actually comparing the received bits to the transmitted bits, these tests verify that the round-trip time is within limits and examine the status registers and end pointers for error indications.

3.3 PSAT/ESI Stateword Protocol Issues

Statewords define the coding and modulation type to be applied on each segment of a burst passed to the ESI by the PSAT. The current PSAT/ESI burst format specified in W-Note 13 is indicated below:

```
                         SYN
                         DLE
                         STX

                  -----------------------------------
        WORD 0    |            (16 bits)            |
                  -----------------------------------
             1    |                                 |
                  -----------------------------------
             2    | N(4 bits) |                     |
                  -----------------------------------
             3    |                .                |
                  |                .                |
             m    |                .                |
                  -----------------------------------
           m+1    |                                 |
                  |          N STATE WORDS          |
           m+N    |                                 |
                  -----------------------------------
                  |                                 |
                  -----------------------------------
                  |               CRC               |
                  -----------------------------------
                  |               CRC               |
                  -----------------------------------
                  |                                 |
                  |                                 |
                  |             TEXT                |
                  |       (Coding by ESI            |
                  |       based on Statewords)      |
                  |                                 |
                  -----------------------------------
```

The two stateword formats are the single state format:

```
-------------------------------------------------------------------
| 0 | B/Q(1) | Code Rate(2) |   State Length in Words(12)         |
-------------------------------------------------------------------
```

and the double state format:

```
-------------------------------------------------------------------
| 1 | DSC(3) | Length State I(4) | Length State I+1(8)            |
-------------------------------------------------------------------
```

where numbers in () specify field lengths in bits, B/Q specifies modulation type, all state lengths are in units of 16-bit words, and DSC specifies a B/Q and Code Rate for each of states I and I+1 based on a table stored in the ESI. Single and double state formats can be mixed freely in a burst control packet. However, since the number of bits in the state word length field (N) is 4, only up to 15 statewords specifying up to 30 states within the text are possible. The following discussion summarizes the problems associated with this design.

3.3.1 Double State Format Shortcomings

The motivation for the double state format is to permit efficient specification of the two states "typically" associated with an application message: header (state I) and text (state I+1). This format, however, does not appear to be ideal for dealing with the traffic which will likely exist in the Wideband Network. In particular, the 4-bit state I field is too small to cover the headers that will be associated with many messages. The Wideband Network itself requires 6 words of message header information. IP has a minimum of 10 header words and TCP also has a minimum of 10 header words. Although there may be less than 16 total header words, this will certainly not be the case in general.

One could think of increasing the size of the double state format state I field at the expense of the state I+1 field. Only being able to represent a text portion limited in length to 2032 bits (i.e. 2**7-1 words) may be too restrictive, however.

An alternative worth considering is to redefine the 4-bit state I field to be an index into a table of lengths rather than being a length itself. The 16 lengths in the table could be the most common total header lengths for the Wideband Network traffic. Headers which do not match one of these standard sizes would be required to use the single state format. At best, however, this approach allows one to specify only up to 15 messages per burst assuming different coding rates for header and text.

A worst-case simplifying assumption for the remainder of this note is that only the single stateword format can be used.

3.3.2 Impact on Datagram Traffic

The PSAT aggregates datagram messages into a single burst to the maximum extent possible in order to improve Wideband channel efficiency. Packets/messages are aggregated as available at each symbol rate subject to the maximum burst length constraint of 2**14-1 channel symbols (2**15-1 bits for QPSK). The fixed number of statewords per burst, however, imposes an additional constraint on the amount of aggregation which can be carried out. The severity of the stateword constraint depends on how the coding is defined on the aggregated packets. If we assume that a "typical case" is all headers coded at rate 1/2 and data coded for less protection, we will need 2 statewords per packet which limits us to 7 packets aggregated in each burst.

Since the PSAT is currently set up to transmit only one datagram reservation per frame, the limit of 7 packets associated with each such reservation could very well imply a serious constraint on the Wideband Network datagram throughput achievable by a PSAT. We already have plans to get around this limitation and expand the PSAT's ability to permit 2 datagram bursts per frame. This limits the PSAT to transmission of 14 "typical" messages per frame steady-state. Expansion to more than 2 datagram bursts per frame, however, will involve a non-trivial modification of the internal PSAT formats. Moreover, one must question the advisability of introducing multiple datagram bursts per PSAT per frame when a single burst will do. In addition to a loss in channel efficiency, multiple bursts increase the processing load on the PSAT due to additional burst scheduling, input and output.

3.3.3 Impact on Stream Traffic

Similar problems arise due to a restriction in the number of statewords in the case of streams. Assume for the moment that the ST protocol is not being used. Hosts create host streams which are aggregated into channel streams by the PSAT. At any channel symbol rate, all host streams originating at a single PSAT with the same interval and phase (i.e. frame scheduling offset in the case of intervals 2, 4 and 8) are combined into a single channel stream. Within a host stream, a host may transmit multiple messages up to a maximum specified at host stream setup time. If stream message headers are to be protected at a higher level than the data portions of the messages (as assumed for datagrams above), a channel stream burst will be able to support only up to 7 stream data messages. Assuming that we allocate statewords as we allocate channel time when a host stream is established, the sum of the maximum messages per host stream slot over all host streams in a channel stream must be less than 7. This means that if any one host stream in a channel stream is set up with a maximum of 7 messages per slot, no other host stream in that channel stream will be able to be subsequently established, even if the first host stream is not originating any transmissions. Moreover, limited stateword availability can restrict host stream throughput below that which the channel is otherwise capable of supporting. If only a single stateword is available for messages of a host stream, host stream throughput is limited to a maximum of 800 Kbit/sec independent of other channel activity (assuming 16,000 bit messages). Smaller message sizes imply a proportionally lower throughput (50 Kbit/sec for 1000 bit messages). The limited number of statewords, which can lead to reduced performance in the case of datagrams, may result in even poorer performance in the case of streams. One can think about introducing multiple bursts per channel stream as described above for the case of datagrams, but the same disadvantages that exist there exist here as well.

The problem is reduced significantly for stream traffic between ST agents. Since ST agents aggregate stream transmissions with all header information collected together in a contiguous portion of the burst, each such "envelope" requires only 1 of the 7 scarce stateword pairs for a given interval and phase. On the other hand, the limit of 7 still restricts the number of other ST agents with which the agent (e.g. Voice Funnel) at a given PSAT can be simultaneously in conversation unless we modify our current burst generation strategy.

3.3.4 Conclusion/Recommendation

It seems apparent that the current number of statewords which are allowed per burst is insufficient to support the patterns of traffic that are likely to arise in the Wideband Network over the next several years. Everyone should be aware that although we can initially bring up the system with the current specification, there are non-trivial limitations which will result from this decision. The maximum allowable number of statewords should be increased significantly in the near future. A PSAT/ESI burst format capable of supporting up to 128 statewords per burst is recommended.

4 REMOTE SITE MAINTENANCE

4.1 System Installations

During this period, on-site system installations were performed at NPS and CINPACFLT. The procedures used at NPS were somewhat different from those which are used conventionally, and are described below.

The proper order is to install the new header files, build the libraries, then the kernel, and finally the programs which depend on kernel tables. After that, the commands which do not depend directly on the kernel tables are constructed.

The "install" command and the "build" install lines assume that the system is dedicated. To build the new kernel "off line" while the system is running timesharing, one needs a compatible way to compile things. The libraries are a problem, but if the libraries are upward compatible, it is relatively safe to install the new libraries in /lib and build a new system underneath the old.

A root is constructed on the small file system, along with the new /bin, /etc, and /etc/net.

One can use the "chroot" facility, if it is available, to handle this, and the new /usr/bin. This was not available in sys.124, so the work on /usr/bin could not be performed as cleanly.

4.2 Version 7 Conversions

4.2.1 Objectives and Constraints

The Version 7 (V7) UNIX environment is slowly becoming available on the BBN-UNIX system. This environment can be divided into two parts:

(1) The programming environment (system calls and libraries), and

(2) The command environment.

The goal is to provide an environment as nearly similar as possible to the standard V7 system, without actually running the V7 kernel. The latter is not feasible at this time because of (1) incompatibilities with V6, and (2) BBN features of the V6 kernel.

Because the kernel is not actually a V7 one, certain differences are unavoidable. The first part of the environment--the programming environment--is nearly complete; the command environment is less so.

4.2.2 Kernel and Library Modifications

The V7 programming environment is simulated by the library /lib/lib7.a. This library is loaded automatically by the cc7 compiler command; it may also be used explicitly with other compilers by specifying -l7 on the command line. The library provides additional functions, and maps between incompatible V6 and V7 calling sequences.

In general, the library ovides the environment described in chapters 2 and 3 of the UNIX 7th Eoition manual. The following differences exist at this time:

(1) The ASSEMBLER portion of each page of manual chapter 2 is usually incorrect. This is not important because the interface programs should always be used.

(2) The system calls differences are in Table 1 below.

(3) Other discrepancies which exist in the library are summarized in Table 2.

The default I/O library is like the libS ("stdio") package provided with the Phototypesetter compiler. A V6 program using stdio will require no work in this area, unless it uses feof for end-of-file processing, in which case it may develop new bugs due to differences in the way the V7 I/O library reads.

chown	silently truncates owner and group to the range 0-255.
dup	the value referred to as "approximately 19" is 40 on BBN-UNIX.
fork	does not fail on inadequate swap space or exceeding limited number of process slots.
fstat	is simulated. st_ctime is a copy of st_mtime, and both are updated when a file's mode, owner, group, or contents is changed. fstat cannot be used to get the status of network file descriptors.
fstat	cannot be used to get the status of network file descriptors.
gtty	returns scrambled bits in its third word if used on non-tty devices. TIOCHPCL, TIOCFLUSH, TIOCGETC, TIOCSETC, TIOCGETP, TIOCSETP, TIOCSETN implemented in the system call interface routine.
lock	not implemented.
lseek	-1 is not returned for seeking to before the beginning of the file.
mount	does not require that the stub upon which a filesys is mounted be a directory.
mpx	is not implemented.
nice	is absolute, the range is not limited.
phys	not implemented.
pipe	the number is 1024, not 4096.
pkon, pkoff	not implemented.
ptrace	request 9 is not implemented. Request 4 always fails on pure-procedure programs.
exec	trap is not implemented.
setuid, setgid	truncate their argument to 0-255.

signal	SIGTERM is 16, not 15; 15 is SIGTNR.
stat	is simulated.
umask	not implemented.
utime	sets the accessed time of the file to the current time.

Table 1. System Call Differences

EDOM, ERANGE	are not defined in math.h (this is a Bell V7 documentation error).
assert	requires stdio.h as well as assert.h.
crypt	does not work as described.
dbm(3X)	is not available.
encrypt	is not available.
mp(3X)	is not available.
pkopen	and related functions are not available.
plot(3X)	is not implemented.
printf	and related functions include an undocumented %r format character (this is true on all V7 systems).
setkey	is not available.
sys_nerr, sys_errlist	should not be used, use errmsg() instead.

Table 2. Other Library Differences

The stat and fstat system calls now return a different structure and use a different include file describing the format, and different member names. Gtty and stty work with a new, somewhat different structure.

4.2.3 Commands

4.2.3.1 Initial Conversions

The most important of the Version 7 utilities which have been converted up to this time are listed in Table 3. This list is not complete, but is intended to be suggestive of the new facilities which have been added.

shell	command interpreter.
[	test program for shell.
expr	expression evaluator for shell.
man	display a manual page.
prman	print a manual page.
manix	search manual index.
getman	get a manual page for editing.
c6to7	convert manual page to Version 7 format.
makeman	make a new or modified manual page available.
adb	debugger.
awk	text processor and report generator.
cc7	Version 7 C compiler.
egrep	fast grep for expressions.
lex	lexical analyzer generator.
lint	C program analyzer.
make	program constructor.
sed	stream editor.
yacc7	parser generator.

Table 3. Version 7 Utilities Available in BBN UNIX

4.2.3.2 The Shell

The Version 7 shell, often called the Bourne Shell, is now available to BBN-UNIX users. It depends on a number of the kernel modifications already implemented at BBN. The shell is described in the Bell UNIX 7th Edition documentation.

The syntax is incompatible with the V6 shell. In order to ensure that shell files written in either syntax are working, a compatibility program has been installed. This program is explained below. It requires that shell files expecting the V7 shell have as their first line ": V7". If the first line of a shell file begins with these four characters, the V7 shell will be executed on it. If it does not, the V6 shell will be invoked.

The version 7 command "test" is installed under the name "[" (open square bracket) only, to avoid conflicts with private test programs. When invoked by the name "[", an argument consisting of just "]" is required in order to indicate the end of the argument list. In all other ways, this command is just like the documented "test" command.

4.2.3.3 Restrictions

Build executes the V6 shell, while make uses the V7 shell exclusively.

There is one problem with trying to determine which shell should be invoked which the compatibility program cannot solve. It arises from the syntax

```
    sh - name
```

Both shells consider this command legal. However, it is impossible to determine for certain whether "name" is the name of a shell file. The standard shell interprets this command as a request to start reading and executing commands from the standard input, with $1 set to "name". The Bourne shell interprets this command as a request to run the shell file "name". The current solution to this problem is to assume that the V6 syntax is being used.

4.2.3.4 Implementation

The compatibility system depends upon a program, installed as "/bin/sh", which decides whether to invoke /bin/sh7 (the new shell) or /bin/sh6 (the old shell, renamed). "/bin/sh" is no longer the shell. The Build.info in /usr/src/cmd/sh has been updated to install all these programs in their correct places.

The environment variable $SHELL is set by login to the name of the shell in the password file entry, or /bin/sh if there is none there. Normally, the vector simply runs that shell; however, if the shell is being invoked on a shell file, the file is examined to determine which shell to run. That decision is propagated down to inferior shells by changing $SHELL, so that (e.g.) a V6 shell file containing a "for" command will invoke the V6 shell inside "for", while a V7 shell file containing a use of "ed" which invokes a subshell to execute some commands will invoke the V7 shell, regardless of the top-level shell employed by the user.

Unfortunately, the technique is not quite as clean as is implied by the statements above. When the V7 shell executes a shell file, it does not (unlike the V6 shell) execute "/bin/sh" to run it; it merely does an internal non-local goto. So the V7 shell on BBN-UNIX has been modified to check whether the first line of a shell file is ": V7", and if so, executes "/bin/sh", which can then perform its function.

One more functional change was made to the V7 shell to accommodate existing V6 software. It only executes .profile when its zeroth argument (argv[0]) consists entirely of "-". The V7 shell executes the .profile whenever its first argument begins with "-".

4.2.3.5 Make and Build

A program to maintain programs significantly aids in systematizing software installation and maintenance. BBN developed "build" to solve these problems but it had drawbacks of its own. It required explicit declaration of all dependencies resulting in large and verbose data files. It lacked a macro facility that would increase readability of the data file. With Version 7 UNIX a similar program, "make", became available. "Make" uses implicit dependencies so that an object file's dependency on its identically named source file does not require explicit declaration. "Make" also offers a simple but effective macro facility. After careful consideration "make" was chosen to replace "build" after incorporation of some of "build"'s best features.

One enhancement to "make" included a special syntax for handling archives. Archive members now implicitly depend on their original source file; a separate copy of the members is not required. The order of the archive's members is defined by their order in the data file.

The other major addition allows a single process to execute all commands instead of creating a separate process for each command. This speeds up execution and simplifies control of sources in different directories. A "build" to "make" data file converter was written to aid in the conversion. Future work on "make" involves path name searching for system files and improved error handling.

4.3 System Control System

4.3.1 Manual Sub-System

The manual sub-system has been used as a test bed for the system control system ideas for some time. The elements in this subsystem include the nroff input pages (source files), the nroff output pages (object files), and the manual index (derived data). In the past, the manual page retrieval program "man" reformatted the command line information to identify the correct output page to display. While this works reasonably well for commands in the original Version 6 distribution, the later manual pages are far more likely to be composite. In fact, in the manual revisions undertaken since the introduction of the index, BBN has combined a number of pages.

The newest version of the "man" command, which is written in the language of the Version 7 shell, uses information from the index to identify manual pages. The index entries have a new field derived from the pathname of the input file. This field is used for inserting and deleting index entries, and for constructing the tables used to find the pages.

The shift from Version 6 to Version 7 includes a conversion of manual page input format. The new system has rather different macros, and certain other conventions (like the use of Roman numerals to identify manual sections) have been changed. The conversion is fairly straightforward, and a mechanical converter, based on the Version 7 commands "awk" and "sed", is used by the "getman" command when a Version 6 page is called out of the input file system. The version of the page can be distinguished by the filename. If "getman" is used on an old-format manual page, the text in the nroff manual source directory is left untouched.

When "makeman" is used, it will check for an old-format page first, then record the differences and perform any required renaming operations.

The current scheme requires the deferred construction of the actual index (That is, both a source and object file exist for the derived information). It may be possible to perform this transformation at installation time. The shell files "man", "getman", and "prman" share a significant amount of code. Also, the "uniq_index" maintenance command is duplicated in manix. Sharing common code would be desirable; it may also be reasonable to consider conversion to a more efficient C language program.

4.3.2 Role of Databases

A workable system for software configuration control is a key component in any overall solution of the remote maintenance problem. Furthermore, software configuration control is essentially a classic database management problem, and many of the approaches and techniques developed by database practitioners are relevant in the domain of software control. Those aspects of software configuration management which give it its database flavor are outlined below. An explicit description of the analogy between software control and classical database management shows how one may apply techniques from database science to this problem.

1. A software control system must keep track of many different kinds of data.

Pure source code control is not enough; it is also important to maintain information on run files ("makefile", "Build.info", and command files), object files, documentation, installation sequences, system change history, sites, etc. Database management systems have been applied in just such situations where there is a multiplicity of types of data.

2. There are significant relationships among the various types of data.

Object files and run files are produced from source files. Manual pages describe commands embodied in installed object files. Part of a site description is a list of software which the user can access. These relationships must be captured and maintained along with the primary data. Database systems provide a framework for representing both information about discrete entities and sets of relationships among those entities.

3. The amount of data can be very large.

For example (see QTR18) BBN-UNIX has 153 kernel source files, 400 historical versions of the operating system, and several hundred user-level commands built from thousands of source files. The original paper on the Bell Labs Source Code Control System (Rochkind, IEEE Transactions on Software Engineering, SE-1, No. 4, Dec. 1975) describes an application involving 100 programmers working on 3000 software modules with an average of 5 revisions per module over a period of 2 years. When the amount of data gets this large, casual methods of information management break down due to either space or time inefficiencies. For example, it is no longer practical to keep separate complete files to represent multiple versions of a source module or document; instead an approach based on a single version plus changes appears necessary. To take another example, consider use of the UNIX "find" command for locating all source files with a given set of attributes. As the size of the software configuration increases, the processing overhead involved in using "find" becomes prohibitive and other methods must be developed for selection of files. Database management systems tackle the problems of large data sizes head-on and can be expected to offer valuable guidance in the areas of data compaction and indexed retrieval.

4. Effective software configuration management depends on the elimination of redundantly stored information.

Some of the most serious difficulties in software control arise when the same (or derivable) information is kept separately in several different locations. This leads to the familiar phenomenon of a bug fix being made in one copy of the source code, but not being propagated to all the other copies of the code, so that eventually the fix disappears when one of the other copies is eventually used as the master for distribution. A subtler case occurs in the maintenance of information describing relationships, or dependencies, of software components (e.g. source file X includes header file A, or run file Y is constructed by loading object files B,C,D). Ideally this dependency information should be represented once and once only, in some central, controlled location. In the current RSM system, the dependency information is spread through a series of "Build.info" and "make" files which are controlled by informal procedures. These can easily drift out-of-date with respect to other aspects of the software configuration. It might, therefore, be advantageous to construct a centralized repository for the dependencies and to automatically generate the appropriate "build" or "make" files as needed. Again, database theory places heavy emphasis on the elimination of redundant storage of data supported by specialized tools for multi-site data distribution (as in SDD-1, work performed for ACCAT by Computer Corporation of America) .

5. As much as possible, it is desirable that objects be identified by their contents and attributes rather than by their location.

UNIX system maintainers spend a significant amount of time deciding where to place things, and remembering where things have been placed, within the file system hierarchy. They may consider, for example, whether to place a new software module in an existing directory or whether to create a new directory to contain it and (perhaps) associated other files. Although it is not always viewed in these terms, the choice is often, ultimately, an efficiency decision. This is because the UNIX directory-search mechanisms become inefficient when confronted with too many files, and UNIX programmers have learned that it is better to create deeply nested directory trees with fewer files in each directory. The result is that one must know the address of a file in rather considerable detail ("/usr/src/cmd/as/as1.c") in order to find it. If better retrieval mechanisms were available, a much flatter view of the world would be possible. Since the location of a file is really an uninteresting detail, this would be preferable. This discussion parallels the current debate between proponents of relational database systems and proponents of hierarchical or network systems. In general, the trend has been toward relational systems with their flat files and non-navigational access methods. These systems allow one to concentrate on what the data is rather than where it is.

6. A complete software control system would include convenient facilities for structured data entry, ad-hoc queries, and report generation.

Since configuration management is, after all, a management function, there is a requirement for friendly administrative tools such as the ones found in a modern database management system. For example, a forms-based front-end for entering trouble reports or for describing new software modules might be useful. An interactive query language could be used to pose questions like, "What commands have had source code changes within the past month?" or "Which source files reference types.h as an include file?" A report generator utility closely coupled to the database representation could assist in the production of configuration status reports.

In light of the above-described similarities between the software configuration management problem and the problems traditionally addressed by database management systems, one might hope to apply an existing database system directly to software configuration management. The problem, however, is that few database systems are comfortable dealing with large blocks of text (like source files and manual pages) as fundamental data items. One class of systems, known as "document retrieval systems" (which help keep track of memos and reports), may offer techniques that can be directly applied to the problem.

5 INTERNET DEVELOPMENT

5.1 VAN Cateway

The VAN gateway represents a new facility for the internet community, but also introduces a new mechanism whose failure or misuse can seriously affect the system. The key problem with use of the VAN gateway, for example to connect the ARPANET and Telenet, is to allow and encourage using it for legitimate purposes, while preventing utilization by unauthorized users or as a result of a software or hardware failure in the networks involved.

There are two aspects to thi. problem. The first control on the gateway usage must be to assure that the packets being handled are legitimate, in that they are associated with authorized users.

The second control problem is to assure that the gateway is being used as intended, with a reasonable level of traffic for the function being performed. Even if packets are being processed for authorized users, it is possible for failures within the routing system, for example, to cause packets to loop endlessly. Failures of the network protocols could similarly cause duplicate packets to be sent needlessly.

In both cases, the concern results from the highly visible impact which use of Telenet incurs, since charges are computed on a per-packet basis. However, the same issues are inherent in the Catenet itself, in that misuse of the network consumes resources which are then unavailable for legitimate use. Thus the problem of managing use of the gateway is most critical for the VAN gateway, but applies as well to all gateways, and in fact to any shared resource.

In the initial implementation of the VAN gateway, resource management is being provided by use of tables which enumerate the authorized users of the gateway. These users are simply the addresses of the hosts, both on the ARPANET (Catenet) side and on the Telenet side, which will be acceptable as valid source and destination addresses of packets which transit the gateway. All other packets which are received by the gateway will be discarded.

In the internet architecture, the Telenet side of the gateway appears as a single network to the internet mechanisms. The gateway contains a table which maps artificial host addresses on that network into real 14-digit Telenet/X.25 addresses, in much the same way as other networks convert internet addresses into addresses for their particular attached network. X.25 virtual circuits are only permitted between the gateway and X.25 hosts which are present in this translation table, which effectively defines the set of authorized gateway users in the X.25 community.

No similar table is necessary for translation of addresses on the ARPANET side of the gateway, since this translation is well defined by the internet protocol. Without any additional mechanisms, this would permit any ARPANET host to use the VAN gateway. In addition, since gateways to other networks are simply ARPANET hosts, this would permit any host on the Catenet to use the VAN gateway.

To restrict the user community of the VAN gateway, a second table is provided, which enumerates all internet addresses which are acceptable as sources or destinations on the ARPANET side of the gateway. Each internet datagram which arrives from the ARPANET or Telenet is checked to assure that the source and destination addresses in the internet header are listed in one of the two tables which define the set of hosts which are permitted to use the VAN gateway.

The table entries will be set up directly by DARPA. In selecting the set of valid hosts, the reliability of the data presented to the gateway should be considered.

In particular, we note that there is a significant difference in the addresses presented at the gateway in the internet header of each packet. If such an address is in fact on the ARPANET, the gateway can verify it by comparison with the address supplied by the IMP in the ARPANET leader of the packet. For packets sent to the ARPANET, one can similarly expect the IMP subnet to deliver the packet to the host specified in the ARPANET leader.

If the address of a packet handled by the gateway is on another component network of the Catenet, the packet is necessarily handled through one or more gateways. The internet structure permits gateways to freely enter the system. Gateways are in general under the control of the organization which owns them and/or the attached network. Gateways in general do not check the addresses in the internet headers of packets which they process, so it is possible for malfunctioning hardware or software to emit packets with incorrect addresses. If such addresses happen to be present in the VAN gateway tables, these packets will be processed by the VAN gateway.

The impact of this situation on the policy for allowing use of the VAN gateway is that hosts on networks other than the ARPANET are to be considered somewhat less reliable in terms of enforcement of the usage policy. The mechanisms in the initial VAN gateway implementation will provide some degree of control over the use of the gateway, but these mechanisms are not to be considered appropriate or complete in the general sense, and they are not proof against failures. These mechanisms are intended only as an interim measure.

We suggest that further development work on the internet gateway system, of which the VAN gateway is a component, should address the problems of resource control at the internet system level. Any mechanism which restricts the usage of a gateway must be designed in conjunction with other network mechanisms, such as routing, flow control, load sharing, and error control.

As an example, we can consider a hypothetical configuration in which two VAN gateways are connected to Telenet, one from ARPANET, and the other from SATNET, to support traffic between Telenet users and hosts on ARPANET or SATNET. Only these user/hosts would be listed in the VAN gateway tables.

Since the VAN gateways are participants in the internet routing mechanisms, failure of the gateway between ARPANET and SATNET would cause the system to recognize the path through TELENET, as a transit network, as a viable route for ARPANET-SATNET traffic. However, this traffic would be discarded at the VAN gateway because the addresses are not listed in its tables.

This scenario will be avoided by preventing the two VAN gateways from being "neighbors" for routing purposes, which is acceptable only in restricted configurations. The general problem results from the usage restrictions at the VAN gateway, which makes the path a valid route for one class of packets, but invalid for other classes. The current routing mechanism cannot handle this situation.

In addition, the effect of the usage restrictions on future mechanisms for handling partitioned networks, and load sharing of gateway paths, must be investigated.

We have two mechanisms to propose for consideration as mechanisms to attack this problem. The first, and better developed, is a resource control model which is a result of the TIP Login work. The second involves the use of performance models, which monitor use of resources to determine if unexpected behavior occurs. These two mechanisms can be introduced to work more effectively in the VAN gateway problem.

The architecture for the TIP Login system identifies several abstract modules which interact to implement the resource control functions. A "Control Point" is the module which directly controls the use of a resource. It is responsible for detecting an attempt to use some resource, collecting such information which identifies who is trying to use the resource and what they are trying to do, and then permitting or denying use of the resource. The decision concerning whether or not a particular usage is allowed is made by a "Decision Module." This module takes the information supplied by the Control Point, and applies the algorithm which defines the resource usage policy. Typically, and particularly in the Tip Login case, the Decision Module will obtain more information about the particular user and/or resource involved in the decision, by using a distributed database system.

In the TIP Login system, the Control Point is at each TIP. Decision Modules are located in special-purpose hosts. The database system is present in those hosts as well as on larger database-maintenance hosts, where tools to manipulate and modify the database exist. Typically the Decision Module identifies the particular individual attempting to use a TIP, and retrieves a record of information specific to that individual, which defines his authorizations (or lack thereof).

Much of this mechanism should prove to be useful as a basis for control of gateways as well. In such a system, the control points would be at the gateways. Decision Modules might also be at the gateways, if decisions must be made on each packet. Decisions might be based on source or destination addresses, or on the identify of the individual responsible for the packet. We suggest that this approach be considered for further research.

A problem which is not being addressed in either the gateway or TIP Login efforts is the second control problem mentioned earlier, namely the monitoring of the use of some resource by an authorized user, to guarartee that the resource is being used as intended. In the VAN gateway case, for example, a malfunctioning TCP might cause many unnecessary packets to be handled, but since they are associated with authorized addresses, no control is applied. In addition to the obvious cost and performance penalties, lack of monitoring precludes the use of policies which grant limited use of resources to, for example, allow some users to handle only low-throughput traffic, or low priority traffic. We believe that the use of performance models, embedded within the gateways and for hosts, is a promising direction for attacking this problem.

Limitations of the current LSI-11 implementation of the VAN gateway are likely to preclude any significant testing of these approaches. We have been pursuing these ideas as research issues, which have surfaced during the current implementation efforts.

5.2 CMCC Development

During the past quarter we worked on assembling tools for performance measurements, conversion of the CMCC to a UNIX-based system, and Catenet fault analysis.

A traffic generation and measurement system has been developed at BBN for another project. We worked with the developers of this system to produce a more generally useful system that we could use for Internet performance measures. The existing generator for this system needs a dedicated LSI-11 machine to run on. Machines which can be used in this way a. hard to find, so we also defined a subset of the facilities which allow a much smaller and simpler implementation that could run on a timesharing system, for example. We hope that this will make it easy to have a number of controllable traffic generators distributed throughout the Catenet. We have located a small number of machines that will run the full generator, but their availability has been limited.

The performance measurement message types described in the last quarterly report have not yet been implemented in any gateway, so we have not been able to use them.

The CMCC now runs on a TOPS-20 and is written in Pascal. We are in the process of converting it to a UNIX-based system in order to integrate it with a new network monitoring system being developed at BBN. The conversion has two main components: translating the code from Pascal to the C language, and splitting it up into a number of small processes. Currently it consists of two large processes, but it cannot run this way on Unix because of the limited process size available. We obtained a Pascal-to-C translation system to assist in the first part of the task, and the second part is now done.

We spent some time tracking down a problem which caused packets to loop between a port expander and a gateway. This was first noticed as abnormally high CMCC traffic counts from one of the SRI packet radio gateways. Tracking down the problem pointed out the need for obtaining more information from gateways remotely: a major step in solving the problem involved getting hold of information that was only available at the terminal attached to the gateway. The problem was eventually traced to port expander behavior and was corrected by SRI.

6 MOBILE ACCESS TERMINAL NETWORK

6.1 Summary of Past Quarter's Work

As part of our participation in the development of the Mobile Access Terminal (MAT) and the MAT Satellite Network (MATNET), we were finally able to ship two complete Red subsystems to E-systems, ECI Division, in St. Petersburg, Florida, for systems integration and preliminary testing. Late deliveries of the BBN C/30 packet switch processors to the MATNET project delayed actual shipment of the equipment by several months. Before taking delivery of the C/30 equipment, we had to share BBN backroom development machines with other C/30 users, which not only was inconvenient but also required extra time to set up the machine to a MAT configuration each session. During the last quarter, we also finished the design and implementation of the software and firmware for the MAT C/30 Satellite IMP.

The shipment to ECI, one complete Red subsystem for a shipboard MAT and one complete Red subsystem for a shore-based MAT, included the following:

QUANTITY	EQUIPMENT
2	BBN C/30 packet switch processor
2	Cassette tape drive
2	DEC LSI-11/03 minicomputer
3	TI-743 hard-copy terminal
1	HP 2648A graphics terminal
2	Racal Vadic VA 3451 1200 baud modem
2	Racal Vadic data phone
	assorted cables

The Racal Vadic phones and modems in the above list are leased items for system checkout at ECI only. Except for the cable between the cassette loading device and the C/30, which we overlooked in packing, all the equipment arrived at ECI as scheduled; we shipped the cable separately, so that it arrived 18 hours later through overnight delivery. The only casualty in the shipment was a single LED bit indicator broken off the BBN C/30 universal I/O board; the LED has subsequently been replaced. BBN personnel were present at ECI to unpack the equipment and to install it into the Red subsystem racks. Upon initial checkout, the C/30 ran CPU and memory diagnostics with no problems. The LSI-11/03 minicomputer, the HP 2648A graphics terminal, the TI-743 terminals, and the modems also functioned correctly.

Although preliminary tests of the control lines and the data lines in the Red/Black interface were successful, further tests revealed some unexpected problems with the software handling the satellite channel I/O and the VDH terrestrial channel I/O. We found and corrected a buffer management problem which resulted in all available buffers being allocated, leaving none free.

(Without buffers, the Red processor was unable to receive channel information to establish frame synchronization.) Subsequently the software appears to be running successfully with only a few bugs currently remaining.

During testing, the leased Biomation Digital Logic Analyzer failed twice and had to be replaced. Another ancillary problem requiring our debugging time to unravel was that the Racal Vadic modems between the C/30 and the BBN PTIP failed to work at 1200 baud unless two stop bits per character were generated. This is not standard procedure with most terminals.

An abnormally large variance in the received-packet arrival time signal initially prevented the C/30 from achieving stable operation with the Black processor. A patch in the Black processor to reduce the buffer size decreased the variance to an acceptable level and permitted full MAT operation.

During this quarter, BBN and ECI together passed the first major milestone in the project, namely the successful transmission of packets through a 25 KHz transponder in a FLTSAT satellite. While satellite time was available, packet error rates for various carrier levels were measured.

6.2 Microcode Design

The BBN C/30 MATNET specific microcode was implemented as changes to the existing microcode for C/30 emulation of the Honeywell 316 general purpose computer. Major changes were made in only two modules: the modem interface I/O emulator and the instruction set emulator. The changes to the former were required to create a Red/Black satellite channel interface, while the changes to the latter were required to add interrupt channels for the new timing instructions. A few additional commands were also included in the console command interpreter.

The SATNET satellite channel I/O interface, on which the MATNET interface is based, incorporates a 10-millisecond clock, a register for holding the transmit-packet start time, a register for recording the received-packet arrival time, and a register for holding a wakeup timer. Of these registers, only the clock and the received-packet arrival time are read by the CPU. In the MATNET satellite channel I/O interface, registers are similarly used to hold times of incoming events and to control the times of the transitions of the control lines by which the macrocode software controls the outgoing events. Each of these timing events is provided with an instruction by which macrocode can either read the time (of input events) or set the event time. This is similar to the SATNET satellite channel interface except that there are more events. Each of the events is provided with a separate interrupt which informs the macrocode of occurrence or completion, including the wakeup event. This is in contrast with the SATNET satellite channel interface, where the wakeup event shares the transmit DMA interrupt.

In the MATNET Satellite IMP, the incoming events are received-packet start, which does not require a special interrupt, and transition on the Unique Word (UW) line, which is polled by the C/30 microcode. Outgoing events are the timed transitions of each of the three control lines: PTI, TPA, and GOSIG. In order to preserve as much Satellite IMP macrocode as possible, event times are still specified by the macrocode in units of 10 milliseconds, although the C/30 is able to provide only 100 millisecond hardware resolution.

6.2.1 Interrupt Architecture

The Honeywell 316 architecture provides a 16-bit register, which holds the state of the 16 priority interrupts. In the standard configuration, all interrupts are already assigned to existing host and modem lines. Although some of these interrupts might be reallocated for MATNET use, this would reduce the number of hosts and land lines the MATNET Satellite IMP could support and would present compatibility problems. Instead, a second interrupt register providing another 16 channels was implemented. These are accessed by the instruction SET MASK (SMK) 121. (The original 16 interrupts are accessed by the instruction SMK 120.) The SMK instruction is encoded as an OTA instruction, referencing a device which is the register holding the mask of the interrupt channels.

An alternative would be to implement a second bank of interrupts with the instruction SMK 220, i.e. the same device code, 20, but a different control code In this scheme, the interrupt vector assignments would have been the next 16 sequential locations in page 0 of main memory after the last of the existing priority interrupts. Locations in low core, however, are at a premium, and the existing Satellite IMP macrocode uses all the ones immediately following the present 16 interrupt vectors. Thus, the choice was made to place the second bank of interrupts at locations 750 to 770 in octal and to implement a new device code. (The microcode references 750 plus the channel number instead of 63 plus the channel number.) A new device code was used because the C/30's I/O hardware provides a convenient mechanism for selecting a base value to address high speed registers as a function of the device code in an I/O instruction.

The priority interrupt dispatching microcode now executes with one of two values set into the C/30's base register. The selected bank of eight registers contains the mask and interrupt request bits and the dispatch address for channel 0 plus locations which are used for subroutine calling. The block of registers corresponding to device 20 also holds information about the non-priority interrupts that are part of the Honeywell 316 architecture, including, for example, the console device's interrupts. In device 21's block of registers, these locations are unused.

In implementation, routine SETREQ120 was modified to provide two entry points, SETREQ120 and SETREQ121, which are used by microcode to schedule macrointerrupts in one of the sets of interrupt channels. All existing I/O interrupts call SETREQ120 as before, while new interrupts call SETREQ121. The microcode calling the SET REQUEST 121 routine closely parallels the macrocode execution of the SMK instruction.

6.2.2 Interrupt Timing

Changes to the I/O microcode can be divided into two parts: the provision for new interrupt timing and polling functions, and the modification of the satellite channel packet formatting microcode. Modification of existing microcode rather than new microcode was indicated, since a MATNET satellite channel modem interface has to emu[illegible]e a standard Honeywell 316 modem interface when its satellite fun[illegible]ions are not enabled. An efficient technique was required so that the new functions did not degrade the performance of the Honeywell 316 emulation when the MATNET satellite channel I/O operations were either enabled or disabled. A general and manageable technique was needed to organize the information related to the several timers.

Basic to the operation of the MATNET Satellite IMP is that interrupt times are specified in 10 millisecond units, while the microcode receives interrupts from the C/30's hardware clock only every 100 milliseconds. Although a facility was available in the C/30's USYS microcode which allowed an interrupt to be scheduled at a specified 100 millisecond interval in the future, use would have required the microcode to divide by 10 frequently. Even though the divide is possibly not at interrupt level, not having hardware divide capability implies the cost of using this facility is prohibitively high. Furthermore, since the Honeywell 316 emulation already made use of this facility, modification of the Honeywell 316 microcode emulator to share the use of the USYS timing facility would have been necessary. Instead, microcode was inserted which incremented a separate timer by 5 upon each 100 millisecond interrupt and performed any other checking necessary before passing control to the USYS timer routine. The USYS facility continues to be used for the noncritical task of maintaining the C/30's three status lights. When MATNET timing or polling functions are not enabled, the cost of this is three 135 nanosecond cycles every 100 milliseconds, which is essentially the minimum possible without rewriting other parts of the C/30's microcode to relocate variables. One cycle is used to address a block of registers holding MATNET-related housekeeping information, one to update the clock, and one to branch to a microcode location whose address is stored in another register. The branch represents all the decision-making, since the actual results are precomputed. Each piece of microcode which makes a change in the state of the MATNET satellite channel interface updates this register in advance, if required.

One of three addresses is kept in this register. When no modem interface is in MATNET satellite channel mode, the address is the USYS timing routine. When UW polling is enabled, the address is the polling routine. When pending events are scheduled and polling is not required, the address is the timer checking routine. When polling is enabled, a second register holds the address of either USYS or the timer checking microcode, as appropriate, for the polling routine to proceed to when it is finished. This is important mainly in the case where the polling routine finds that the UW line has not changed. A similar principle was applied to the timer checking routine, since its cost is dominated by its running cost when no event has fallen due in the last 100 milliseconds. Three additional cycles are required to make this check (assuming that UW polling is disabled). One cycle compares the newly incremented value of the timer with the stored time of the next interrupt, one cycle is required to conditionally branch to USYS, and one cycle is lost after the branch due to the pipelined execution of instructions on the C/30. This lost cycle is used when the branch is not taken, but the cost is dominated by the case when it is taken.

The time of next interrupt is obtained by recomputing a minimum value each time an interrupt is scheduled. Although this microcode is off the most critical execution path, a number of techniques were looked at to implement this efficiently. Even though comparison of the time of a newly scheduled interrupt with the time of the previously scheduled next interrupt time is sufficient when a new interrupt is requested by macrocode, all remaining event times must be examined when an event falls due for determining the next one to check. Information related to an event is kept in a four-word block in the C/30's high speed register file. These blocks are organized into a linked list, which is maintained in schedule order. When a new event is scheduled, the list is searched from its head for the proper location to insert the new block. Then the time of next interrupt is copied from whatever is now the head of the list. Although it was not strictly necessary, an event is placed in the list after other events which have the same time value. Also, as a convenience to the macrocode, an event may be rescheduled when it is already scheduled. This is detected by the event block having a nonzero link word, in which case the block is first removed from the list and then reinserted in its correct position. In any case, it is necessary to clear the link word of an unscheduled event and check it before scheduling in order to assure integrity of the list.

This technique has the advantage of expandability at no cost in additional microcode and of placing the greater computing burden on the non-interrupt microcode, since all order computation is accomplished when entries are placed in the list. After an event occurs, the head of the list is removed, and the time of the next event is copied from the new head of the list to where it can be found rapidly by the initial interrupt timer checking microcode. If there is no next event, the microcode pointers are updated to bypass timer checking. In summary, to improve efficiency, the state of the interface is stored in the form, possibly redundant, which will result in faster execution of the frequently executing paths and faster execution of interrupt servicing microcode at the expense of non-interrupt servicing microcode.

6.2.3 UW Polling

Since the C/30 has no hardware which can convert a transition of any of the USART's input control lines into an interrupt, microcode must perform this function. The tightest possible coding of this function required 9 cycles, including a cycle lost after the branch and a cycle for the final branch when no transition is detected. When a transition is noticed, the status register is read twice for debouncing, and the direction of the transition is examined. If the transition was 0 to 1, the state of the line is stored and the polling routine exits. If the transition is 1 to 0, polling is disabled until macrocode reads the time of this event, a macro interrupt is scheduled, a copy of the MATNET timer word is saved, and a flag is set to indicate it is valid. When polling is enabled, the minimum 9 cycles are executed every 100 milliseconds.

In addition to storing a redundant, precomputed copy of the exit address, two additional techniques were used to produce the 9-cycle minimum execution cycle. First, a copy of the I/O device address is stored in the register block containing other MATNET satellite channel interface information, so that it can be addressed without changing base. Second, while the state of the UW line is stored in a word with a number of flags used by microcode to represent various aspects of the state of the MATNET interface, the bit position used is the same as that in which the state of the DSR status line is reported when I/O status is read from the USART. The transition is detected by computing the "EXCLUSIVE OR" of the I/O status word and the word of flags and looking at the single bit of the result which is meaningful.

6.2.4 Exit Pointers

Two exit addresses are stored to optimize polling and timer checking. They are maintained as follows.

When MATNET functions are enabled by the MATENB instruction, the polling exit is set to point to the USYS timing routine, and the pre-checking exit is set to point to the polling routine. When polling notices a transition, it disables itself off by copying its exit address to the pre-check exit. Polling is subsequently enabled by the MATUW instruction, which reads the saved transition time value, copies the pre-check exit address to the polling exit address, and sets the pre-check exit to point to the UW polling routine.

When an event is scheduled by one of the several MATNET timing macro-instructions, corresponding changes are made to the exit pointers. If the queue of events was empty before this instruction, timer checking is enabled either by setting the polling exit to point to the timer checking routine if polling is currently enabled, or by setting the pre-check exit to point to the timer checking routine if polling is currently disabled. Note that polling is always performed before timer checking, and that timer functions and polling functions are always performed before allowing the USYS timer routine to run. System throughput is the same, however, no matter what the order of execution of these functions.

6.2.5 Event Timer

Each distinct timed action is implemented by a separate microcoded routine. The action corresponding to MATWUP is simply scheduling the macrocode's timed wakeup. Other routines are assertions and denials of interface signals. The block of four registers associated with each event contains four words, containing a link, a time of execution, a microcode address, and an extra data word. The data word is used to hold the length of the signal assertion for events that correspond to a timed assertion of an interface signal. To assert a signal, the appropriate interface bits are written to the I/O device, the address of the signal assertion routine in the block is replaced with the address of the signal denial routine, and a common routine is invoked in which the data word is added to the clock value preliminary to the inclusion of a new event in the event list. The denial routine will clear the interface bits and schedule the macro interrupt, allowing the macrocode to create the sequences of transitions corresponding to the timing diagrams of the MATNET satellite channel interface specification.

The design does not allow creation of arbitrary timing sequences, because dependencies corresponding to curved arrows in timing diagrams between transitions of one signal and a later transition of a different signal are implemented by MATNET specific microcode. The existence of these dependencies corresponds to the fact that the interface operation is not completely open loop, since it is affected by flow control information from the Black processor. The mask bit to set for scheduling of a macro interrupt is also supplied by the routine which completes the action and schedules the interrupt rather than being stored, for example, in the block of registers.

6.2.6 I/O Framing

New I/O and timing related microcode was placed in a separate module of approximately the size of the existing modem interface I/O module. A minimum of instructions were added to the modem interface I/O module to branch to the MATNET satellite channel microcode instead of the normal modem interface microcode, where appropriate, based on a status bit being set representing the fact that the current modem interface is operating in MATNET satellite channel mode ("MATENABLED"). The number of such additions was minimized by the fact that the structure of the modem interface microcode uses a stored microcode routine address to determine where to branch after receiving the device interrupt. All the high frequency branches are controlled by this mechanism. Thus a number of microcode routines which correspond to the new possible intercharacter states in sending of a MATNET format packet were all added to the MATNET I/O specific module. A few pieces of the modem interface

I/O microcode which were formerly executed in line are now executed as microcoded subroutines. The modified modem interface I/O module could be assembled without the MATNET I/O module and would execute correctly, except that some labels would be undefined. These could be defined in a dummy module to point to an error location (which would never be branched to except in the event of a hardware error), could be systematically commented out, or could be conditionally assembled, which our development facilities will support in the near future.

6.2.7 I/O Dispatching

When the MATNET satellite channel interface microcode is included, two adjacent modem interfaces may be defined as being of the MATNET device type instead of the normal device type. In this case, I/O instructions are decoded by MATNET versions of the decoding routines through dispatch tables. (An alternate technique would have been to partially decode the instruction to determine whether it was one of the few MATNET I/O instructions, and only then to continue the MATNET specific instruction decoding.) The implementation chosen favors the MATNET enabled case, since in general one modem interface will be enabled for MATNET satellite channel operation. Modem interfaces not designated as possible MATNET satellite channel interfaces do not pay any overhead cost.

Two tables and two dispatching routines are used; OCP instructions occupy one table and use one routine, while the remaining three types of instructions were encoded in such a way that they can be compressed into a single table and share the other routine. Those specific I/O instructions which are only valid when the MATNET satellite channel interface is enabled have a table entry which returns to the dispatch routine with the I/O routine's address in a register; hence, the dispatch routine can issue an error microtrap to halt the macro machine. Those instructions which are always valid have a branch instruction in the table and do not return to the dispatch routine. The dispatch routine leaves the condition register set to reflect whether or not the modem interface corresponding to the macroinstruction being executed is operating in MATNET mode. Several of the dispatch routines branch immediately to the normal I/O routine if MATNET operation is not in effect. MATENB and MATNOW are exceptions, since it must be possible to enable a modem interface for MATNET operation when none is enabled. Thus the Satellite IMP macrocode can use the MATNOW instruction to search for a modem interface which is able to be placed in MATNET mode.

7 TCP FOR THE HP3000

Major progress was made in the HP3000 TCP project during the last quarter. All of the major protocol software modules were successfully integrated and some extensive tests were made. To date we have been able to run the TCP in loopback mode; the loopback has been placed in the HDH protocol module, which is the lowest level protocol. The loopback has allowed us to run both a primitive throughput tester and a user TELNET program. The tests show that we can make TCP connections and transfer data from files as large as a million bytes without difficulty. In addition to the file transfer we have done some testing of an interactive TELNET connection and found it to be satisfactory.

Part of our effort has been to increase the sophistication of the protocol software. The following section represents some work done toward that end.

7.1 Foreign Host Address and Routing Database

One important function of network protocol software is to maintain a database of foreign host addresses and network routing information. The database contains accurate information about which foreign hosts are accessible to the local host along with the routing information required to transmit messages to these hosts. The information includes the following items:

1. A character string containing the name of the foreign host.

2. A foreign host internet address. This address includes the destination network as well as the destination host's local network address.

3. A list of hosts on the local network which can be used as gateways to the destination network.

7.1.1 Relationship of the Database to the Prc [illegible] Software

The foreign host address database is accessed from three protocol levels: the user protocol level, the 1822 protocol level, and the Internet protocol level.

The user protocol level uses the database to map foreign host name character strings into foreign host internet addresses. This type of mapping is typically done by programs such as TELNET and FTP before they attempt to open a TCP connection to a foreign host. The Internet Address is used as a parameter to the TCPopen command.

At the 1822 protocol level the internet address is mapped into a local network address. This address represents either the destination host or a gateway to the destination host's network. This mapping is done through a network routing table described below.

The Internet protocol layer is responsible for maintaining the network routing table. The table is altered every time a

Routing Update or Destination Unreachable packet is received from a gateway.

7.1.2 Design Criteria for the Foreign Host Address Table

Several important considerations dictate the design of the Foreign Address Host Address table. These considerations are as follows:

1. The database must be relatively simple to maintain. This requires that it be stored as a normal text file which can be modified by the standard text editor.

2. Mapping of a host name character string is a relatively infrequent occurrence in which speed is not overly important.

3. Mapping of the foreign host internet address into a local network address occurs for every 1822 packet sent out over the network. This requires that the operation be as quick and simple as possible.

4. It must be possible to update the network routing portion of the database in response to Internet routing messages. This requires dynamic updating while the TCP is running.

5. It must be possible to add or delete hosts from the database while the TCP operates. This process should involve no more than editing a text file and invoking an update command which makes the new version of the database the current version. All of this should occur with only a momentary delay in the transmission of data.

7.1.3 The Database Structure

The database structure consists of three entities: a master list file, an index table for the master text file, and a routing table.

The master list file contains a list of all accessible hosts and a list of all accessible networks. The host list is in alphabetical order and can include several names for each host. In addition to the alphabetical order there are dividers which separate the file into 26 sections for each letter of the alphabet; the first character in a host's name dictates which section it falls into. Along with the host name there is an octal foreign host internet address. The address includes the foreign host's network address.

The foreign network list contains an entry for every possible foreign network. The entry contains the addresses of all gateway hosts on the local network which may be used to reach the destination host. The gateway addresses are ordered by the their topological distance from the foreign network. The first gateway listed is closest to the foreign network while the last listed is farthest away.

The master file index table is derived from the master file. The table is made up of 26 entries which contain the record number of the first entry of each alphabetic section. The table is stored in computer memory while the TCP operates. The table is created when the TCP is initialized or when an update command is executed. The index table is designed to speed the mapping of a character string name into an internet address by limiting the search for the host entry to a specified portion of the master file.

The routing table is made up of an entry for each of the possible 256 network addresses. Each entry contains either a NULL to indicate that it is unreachable or the local network address of a gateway host. This gateway host represents the start of the best known path to the destination host.

The routing table is also derived from the master list file. It is only created at initialization time and does not change in response to an update command. The table is only modified in response to an Internet Routing message or a Destination Unreachable message from a gateway. The Internet Routing message will contain the new routing information. A Destination Unreachable message causes the master file to be searched f the next best path to the network. If no such path is found, the network is marked as unreachable.

7.1.4 Master List File Sample

The following is an example of a master list file.

```
Host Name      Internet Address
ahost1            30012454
ahost2            20005868
/***** section divider *********
bhost1            39098989
bhost2            49809800
bhost3            19879799
****** section divider *********
chost1            50809870
Net Id          Gateway Host List
1        8909787  9494837 3484742
2        8857494  348489
3        0
4        8909787  3484742
```

8 TCP-TAC

The major work done in the past quarter was writing and testing new Transmission Control Protocol (TCP) code and converting existing H-316 TIP code to work in the TAC.

Code was written to send and receive TCP segments, open, close, and reset TCP connections, and accept TCP listening connections. This ncluded a finite state machine process to initiate the opening, closing, resetting, and timing of the connections. The code was first tested by opening a connection from one TAC port to another; when this was working correctly, it was tested with some real network hosts. The TAC has now successfully established TCP connections with MIT-Multics and ISID.

As part of the TCP testing a problem occurred that may be either a bug in the MIT-Multics TCP or in the TCP specification. The problem has to do with the handling of Resets when one TCP is in the established state. This problem is currently being studied. If it is a real bug, and not a problem in the TAC code, then the appropriate people will be informed.

An Internet Note, IEN-166, was published which described the design of TCP/IP in the TAC.

The original parts of the H-316 TIP code that supported the Multi-Line Terminal Controller (MLC), TELNET Protocol, and NCP

Host-Host protocol have been separated into separate modules. This code is currently being converted to work with the new TAC control structure and buffer system. Part of this work is to make the TELNET and MLC modules work with both the NCP and TCP protocols.

The dispatching on 1822 Host-IMP protocol messages has been improved. This was done in the areas of such control messages as RFNMs, Incompletes, Destination Dead, and Dead Host status.

An extended debugger (DDT) was brought up in the TAC. This DDT was previously used as a package in the IMP, and has many features that the existing one did not. It has made debugging much easier.

9 TCP FOR VAX-UNIX

This section of the QTR covers the VAX-UNIX Networking Support Project, which is an effort to implement the DoD standard host-host protocols, TCP and IP, for the VAX running the University of California at Berkeley (UCB) virtual memory version of UNIX (VM/UNIX). The details of the implementation have been described earlier (in QTR 19), and an expanded description was published during this quarter*. In addition, a talk describing the implementation was presented at USENIX '81 (the semi-annual UNIX User's Group meeting), held in San Francisco and attended by about 800 people.

The past quarter has seen the completion of coding of the initial version of the network software. This includes:

(1) The TCP protocol layer,

(2) The IP protocol layer,

(3) The 1822 protocol layer, including the IMP interface device driver, and

(4) The UNIX file I/O oriented user interface.

Work also included integration of the network modules with the rest of the operating system. The 3BSD version of the VM/UNIX kernel is being used for initial debugging. Integration required minor changes to the UNIX file I/O system call code

*Gurwitz, R., "VAX-UNIX Networking Support Project Implementation Description," IEN 168, January 1981.

(read, write, open, and close), as well as addition of data structures to support network connection control and buffer management.

Testing of the software is now well underway. The TCP has successfully established connections and transmitted data between processes while in loopback mode. An initial working version is expected by late February. A small amount of coding work remains on the raw 1822 level and raw IP level user interfaces. Design of the capability for simultaneous access to multiple networks has been started, as has work on FTP and TELNET for the TCP. A pre-release version of the software was distributed to UCB for their inspection, as they will be the first test site outside BBN.

The period from the time of the availability of the first working version to the date of planned release to test sites (March 15, 1981) will include phase-in of features such as IP option processing and the raw user interfaces, as well as initial performance measurements and tuning. UCB has released a new version of VM/UNIX (4BSD), and it is expected that the network software will be integrated with it, after a period of evaluation.

DISTRIBUTION
[QTR 20]

ARPA
Director (3 copies)
Defense Advanced Research Projects Agency
1400 Wilson Blvd.
Arlington, VA 22209
Attn: Program Manager

R. Kahn
V. Cerf
J. Dietzler
D. Adams

DEFENSE DOCUMENTATION CENTER (12 copies)
Cameron Station
Alexandria, VA 22314

DEFENSE COMMUNICATIONS ENGINEERING CENTER
1850 Wiehle Road
Reston, VA 22090
Attn: Lt. Col. Frank Zimmerman

DEPARTMENT OF DEFENSE
9800 Savage Road
Ft. Meade, MD 20755
R. McFarland R17 (2 copies)
M. Tinto S46 (2 copies)

ELEX3101
Naval Electronics Systems Command
Department of the Navy
Code 3101
Washington, DC 20360
Attn: Barry Hughes

ELEX3301
Naval Electronics Systems Command
Department of the Navy
Code 3301
Washington, DC 20360
J. Machado
F. Deckleman

BOLT BERANEK AND NEWMAN INC.
1701 North Fort Myer Drive
Arlington, VA 22209
E. Wolf

DISTRIBUTION cont'd
[QTR 20]

BOLT BERANEK AND NEWMAN INC.
50 Moulton Street
Cambridge, MA 02138

R. Alter
A. Owen
G. Falk
R. Bressler
A. Lake
J. Robinson
A. McKenzie
F. Heart
P. Santos
R. Brooks
W. Edmond
J. Haverty
D. McNeill
M. Brescia
A. Nemeth
B. Woznick
R. Thomas
R. Koolish
W. Milliken
S. Groff
M. Hoffman
R. Rettberg
W. Mann
P. Carvey
D. Hunt
P. Cudhea
L. Evenchik
D. Flood Page
J. Herman
J. Sax
R. Hinden
G. Ruth
S. Kent
R. Gurwitz
J. Pershing
Library